

## Article

# Self-Organizing Traffic Flow Prediction with an Optimized Deep Belief Network for Internet of Vehicles

Shidrokh Goudarzi <sup>1</sup>, Mohd Nazri Kama <sup>1</sup>, Mohammad Hossein Anisi <sup>2,\*</sup>,  
Seyed Ahmad Soleymani <sup>3</sup> and Faiyaz Doctor <sup>2</sup>

<sup>1</sup> Advanced Informatics School, Universiti Teknologi Malaysia Kuala Lumpur (UTM), Jalan Semarak, Kuala Lumpur 54100, Malaysia; gshidrokh2@live.utm.my (S.G.); mdnazri@utm.my (M.N.K.)

<sup>2</sup> School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK; fdcto@essex.ac.uk

<sup>3</sup> Faculty of Computing, Universiti Teknologi Malaysia Kuala Lumpur (UTM), Skudai, Johor 81310, Malaysia; asseyed4@live.utm.my

\* Correspondence: m.anisi@essex.ac.uk

Received: 18 August 2018; Accepted: 10 October 2018; Published: 15 October 2018



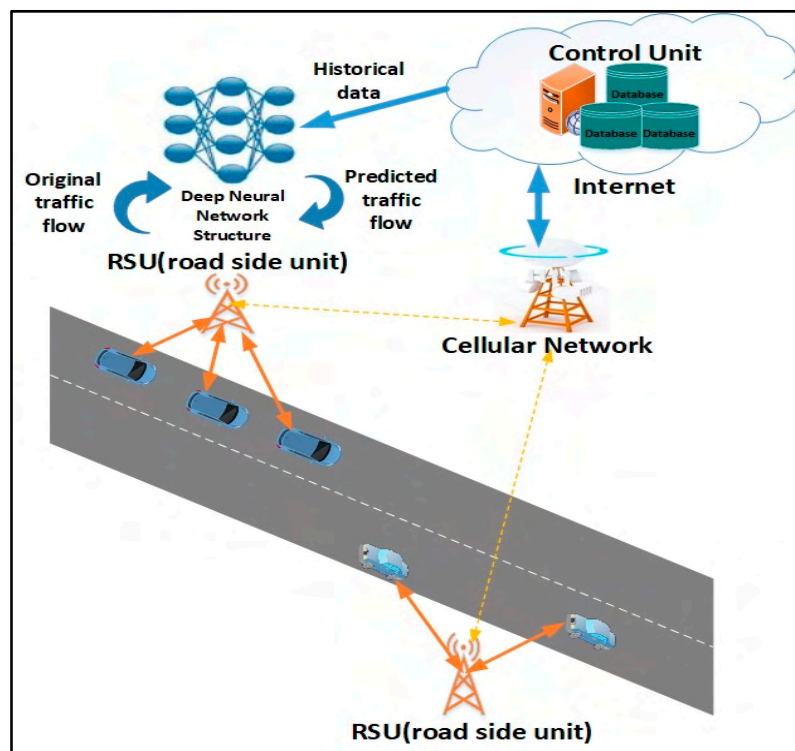
**Abstract:** To assist in the broadcasting of time-critical traffic information in an Internet of Vehicles (IoV) and vehicular sensor networks (VSN), fast network connectivity is needed. Accurate traffic information prediction can improve traffic congestion and operation efficiency, which helps to reduce commute times, noise and carbon emissions. In this study, we present a novel approach for predicting the traffic flow volume by using traffic data in self-organizing vehicular networks. The proposed method is based on using a probabilistic generative neural network techniques called deep belief network (DBN) that includes multiple layers of restricted Boltzmann machine (RBM) auto-encoders. Time series data generated from the roadside units (RSUs) for five highway links are used by a three layer DBN to extract and learn key input features for constructing a model to predict traffic flow. Back-propagation is utilized as a general learning algorithm for fine-tuning the weight parameters among the visible and hidden layers of RBMs. During the training process the firefly algorithm (FFA) is applied for optimizing the DBN topology and learning rate parameter. Monte Carlo simulations are used to assess the accuracy of the prediction model. The results show that the proposed model achieves superior performance accuracy for predicting traffic flow in comparison with other approaches applied in the literature. The proposed approach can help to solve the problem of traffic congestion, and provide guidance and advice for road users and traffic regulators.

**Keywords:** deep belief network; historical time traffic flows; restricted Boltzmann machine; optimization; traffic flow prediction

## 1. Introduction

In an IoV and VSN, vehicles act as senders, receivers and routers to broadcast data to a network or transportation agency as part of an integrated Intelligent Transportation System (ITS) [1]. The collected data can be used for traffic flow prediction to ensure safe, free-flow of traffic in metropolitan areas. The application of sensor networks as a roadside communication infrastructure is regularly used in various current intelligent transportation and smart highway systems. The roadside units (RSUs) offer a secure infrastructure along the road which are responsible for broadcasting periodic safety messages to road users. Typically, RSUs are located every 300 m to 1 km and transmit data at the interval of every 300 ms. Therefore, placing RSUs along a long stretch of highway to offer ubiquitous connectivity is not economically viable. Hence, vehicles should be able to use other vehicles to transmit and receive

driver critical data feeds with limited support from fixed road side infrastructures [1,2]. In this paper, we developed smart prediction scheme for vehicle-to-vehicle (V2V) communication [3,4], where the vehicles can obtain predicted information using their on-board units (OBUs) which is computed by RSUs. A basic scenario of content delivery to vehicles at various ranges through vehicle-to-roadside (V2R) links is presented in Figure 1. The main operational functions of the real time prediction system depicted includes: traffic data archiving, traffic pattern processing and traffic flow forecasting. Traffic data would be collected by RSUs for purposes of data analysis. Traffic pattern processing would create a dynamic traffic pattern (TP) matrix using the collected data to assess traffic volume. This paper focuses on developing the traffic flow forecasting unit which uses the TP matrix for constructing a traffic flow prediction model.



**Figure 1.** Delivery of content to vehicles via vehicle-to-roadside (v2r) links.

The main challenge is that short-term traffic estimations may be inaccurate because of unpredictable disruptions such as accidents on the road. Historical traffic flow data should be used for traffic time estimation in a network. Nevertheless, activity time forecast cannot exclusively depend on past movement information because of the following reasons: (1) On-street disruptions and accidents which would affect traffic flows in the network, the impact of which cannot be anticipated; (2) off-road events can affect traffic flows and they cannot be incorporated into the typical historical traffic time information; and finally, (3) traffic information is not accessible for all connections in a traffic network due to the fact that most connections are not equipped with traffic sensors.

Accurately anticipating traffic time is an imperative element of IoV and intelligent transportation frameworks [5,6]. There are a wide range of traffic time prediction techniques incorporating time arrangement examination [7,8], Bayesian systems [9], neural networks (NNs) [10–13], fuzzy systems [2], fuzzy NNs [14,15], nonparametric regression (NP) [16,17], and other computational intelligence approaches [18]. The availability of travel time data is increasingly being used for modelling traffic behaviour to assist road users and city authorities to make better informed decisions about travel choices, levels of pollution and congestion, the effect on public and private transportation policies, and effective repair and maintenance of the road network. However, data can often be missing for

specific timeframes due to noise in the reading or corrupted data [19–22]. Various machine learning, probabilistic and statistical modelling approaches have attempted to solve the problem of missing data in traffic forecasting [23–29]. A study by van Lint et al. [30] showed a travel time forecasting model based on a neural system for handling missing traffic information while in Sun et al. [31] traffic streams estimation based on using a Bayesian model was presented where missing historical traffic information was estimated by utilizing a Gaussian blend display to visually verify the traffic data forecast. Various specialists have shown that hybrid methods have better results in terms of accuracy and precision compared with individual techniques [32]. Hybrid methods based on fuzzy logic can be potential alternatives to enhance precision in traffic flow prediction as described in [33] while in [34] a novel method based on neural networks is utilized in traffic time estimation.

Artificial neural networks (ANNs) have been widely used for time series prediction problems since their inception in the 1980s. In classical neural networks, training algorithms akin to back-propagation only try to model the dependence of the output from the input. Restricted Boltzmann Machines (RBMs), instead, are networks of stochastic neurons that can be trained in a greedy fashion. Deep belief networks are obtained by stacking RBMs on one another so that the input to one layer is given by the hidden units of the adjacent layer, as if they were data, and adding a last discriminative layer. The RBM, might even yield better results than traditional neural networks with higher accuracy. In a RBN, the hidden units are independent given the visible states. So, they can quickly get an unbiased sample from the posterior distribution when given a data-vector. This is a big advantage over direct belief nets. The multi-layer perceptron (MLP) and radial basis function networks (RBFN) are well-known approaches. Often gradient descent methods are used for training these approaches and back propagation (BP) is used as the learning algorithm [27].

However, there are some limitations of using conventionally shallow ANNs for real world problems such as traffic flow prediction in highways based on VANET-cellular systems. The first issue is related to the design of the ANN topology. It is found that the larger the size of the hidden layer the more prone the model is to overfitting the training data. The second problem is related to deciding the initial value of the ANN weights. BP is a supervised learning method which uses samples of input and output data to modify weights of connections between units (neurons) across the network layers. The appropriate selection of initial weights can increase the speed with which the model is able to converge. Both these problems are amplified when the input parameter space is very large as in the case of traffic flow prediction. Hence there is a need to be able to transform the input parameters into a reduced and manageable feature space with which to construct the prediction model. Equally there is a need to determine the optimal number of hidden neurons for training the model. Finally, the third problem is determining a suitable learning rate during the models training phase. Here there is a need to incorporate an automated way of selecting the most appropriate learning rates as the model is being trained. To solve these problems, we proposed a novel traffic flow prediction model based on DBNs comprised of multiple stacked restricted Boltzmann machine (RBM) auto-encoders. RBMs are networks of stochastic units with undirected interactions between pairs of visible and hidden units which can be used to learn a probability distribution over its set of inputs. By stacking multiple RBMs onto one another DBNs are trained using greedy layer-wise learning which aims to train each layer of a DBN in a sequential and unsupervised way, feeding lower layer results to the upper layers to capture a representational hierarchy of relationships within the training data [10–13]. Each trained layer represents feature encoders which can be helpful in the discrimination of the target output space. This unsupervised training process can provide an optimal start for supervised training as well as extract and learn a reduced set of features representing the input parameters. Supervised learning is then performed using backpropagation for fine-tuning the weight parameters among the visible and hidden layers of RBMs for training the traffic flow prediction model. The firefly Algorithm (FFA) is further applied for selecting the optimal number of connected units (neurons) and learning rate during training of the proposed model which has been termed DRBM-FFA. In brief, the main contribution of this study can be listed as follows:

- We define a dynamic traffic pattern matrix to assess traffic volume data;
- We propose a 3-layer DBN composed of two RBMs to determine the salient features from time series traffic volume data for constructing a traffic flow prediction model on VANET-cellular systems.
- We utilize FFA algorithm to optimize and select the sizes of the learning rates in neural networks and;
- We perform simulations and explain how to use historical traffic data for traffic volume prediction.

The reminder of this paper is organized as follows: Section 2 shows the initial Traffic Pattern (TP) matrix to assess traffic time data at five highway links. A dynamic (TP) matrix predictor based on a DBN of RBMs is presented in Section 3. The (FFA) algorithm for selecting the best number of units and for selecting the rates of learning of deep belief nets is explained in Section 4. We demonstrate our predictions and results in Section 5 and the conclusions in Section 6.

## 2. Assessing Traffic Pattern Matrix

This section focuses on the effective procedure to predict traffic pattern in vehicular communications for utilization in real-time applications, such as dynamic traffic management. RSUs can collect speed and flow data and the information gathered can be delivered to a control unit that automatically estimates volume of traffic [35].

The pattern of traffic can be characterized as a matrix on a temporal and spatial scale. The spatial scale incorporates the entire area of the street for which specific trip times can be anticipated. The temporal scale incorporates adequate time spans to characterize the impact of traffic on travel time. Traffic volume is specified as the number of vehicles that cross a section of road per unit time within a selected period. Volume of traffic can influence travel time together with speed of vehicles which is utilized as a marker for congestion. We assigned the weights at given times and locations to create the TP matrix based on congestion level to optimize travel times. The principal task here is to derive a historical days' database by using the assumption that traffic patterns are repetitive during a tight time period, for example, traffic time for 10 a.m. traffic can be viewed from 9 a.m. to 11 a.m. This search window can locate comparative traffic patterns rapidly. There are traffic examples of different days which are recorded in the database inside a time span of  $\pm x$  minutes for time estimation. Our scenario links to V2R communications and measures vehicles moving at speeds of 100 km/h ( $\sim 27$  m/s) crossing each of the RSU with a coverage range of 200 m (radius). This relates to the high contact duration of  $200 \times 2/27 \approx 15$  s.

In our simulation, we assumed that the road section consists of  $k$  links and each link shows a section of road. Each section should be equipped with one RSU, the amount of days in historical database is denoted by  $i = 1, 2, 3, \dots, n_i$ ,  $j = 0, 5, 10, \dots, n_j$  representing information in a five minutes resolution, and  $t$  which is the prediction time on prediction day  $p$ . The start time of the traffic pattern on historical days denoted by  $t_s$ .  $v(i, t - j, p)$  designates velocity on prediction day  $p$  at link  $k$  at time  $t - j$ . Similarity,  $h = 1, 2, 3, \dots, n_h$  shows the number of days in historical database and  $t_s$  shows the beginning time of the traffic pattern on historical days then  $v(i, t_s - j, h)$  shows on historical day  $h$  at link  $k$  at time  $t_s - j$ . Travel time on a road is mainly affected by the congestion present on the road. This congestion may occur due to bottlenecks. Weights are applied to account for the congestion produced due to the type of bottlenecks, whenever and wherever it occurs. We set the weights according to the rapid speed of each section. These weights have to be higher for the sections with lower rapid speeds which represented bottlenecks. The following practical formula is utilized in Equation (1):

$$w(i, j) = \frac{1}{[v(i, t - j, p)]^C} \quad (1)$$

where,  $C$  is a constant. The search is executed in  $\pm x$  minutes of estimation time  $t$  on historical days so  $t + x \geq t_s \geq t - x$ . The basic purpose of the pattern matching process is to find the most similar historical pattern(s). Hence, the primary task is to generate some historical days' database. One way of

searching these patterns is to discover the entire historical database for the most similar pattern, but this makes the search process computationally intensive. Hence, the sum of the squared difference between the prediction time traffic pattern and the historical traffic patterns is used as a criterion for finding similarities between the traffic patterns. The historical traffic pattern having minimum sum of squared difference, is regarded as the most similar pattern. The objective function formula for forming the traffic patterns can be determined by Equation (2):

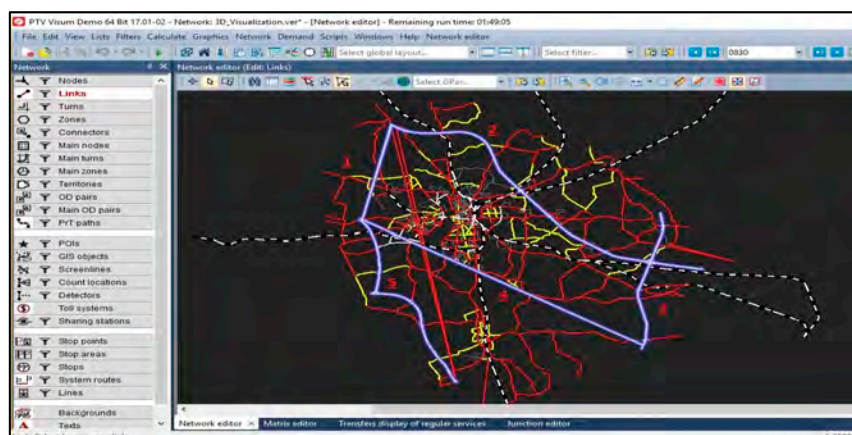
$$\Delta^2(p, t, h, t_s) = \sum_{i=0}^{n_i} \sum_{j=0}^{n_j} w(i, j) \cdot \frac{L(i)}{L} \left[ \frac{1}{v(i, t - j, p)} - \frac{1}{v(i, t_s - j, h)} \right]^2 \quad (2)$$

where the traffic weight in cell  $(i, j)$  is shown by  $w(i, j)$ , length of section  $i$  is presented by  $L(i)$  and the stretch length of the road is shown by  $L$  and  $\Delta^2(p, t, h, t_s)$  denotes the squared difference among the current and historical pattern. After assigning the TP matrix, standard deviation, the coefficient of determination  $R^2$ , the mean square error and linear regression line parameters should be determined. The TP matrix fixes the trip's numbers with zones in each short period of time. Each TP matrix is allocated to each transportation option. Each link shows streets and highways and nodes which can be connected by links. The Table 1 shows values for the highway links.

**Table 1.** Values for highway links.

| Criteria                         | Data                 | Value  |
|----------------------------------|----------------------|--|
| Highway free flow template speed | Raw data<br>120 km/h | Data on 5 mn-spaced intervals<br>10 km/5 mn interval |
| Average link length              | 2 km                 | 5 links traversed/5 mn interval                      |
| Highway Congested template       | Raw data             | Data on 5 mn-spaced intervals                        |
| Average speed                    | 72 km/h              | 6 km/5 mn interval                                   |
| Average link length              | 2 km                 | 3 links traversed/5 mn interval                      |

A commercial software called PTV Visum [36] is used to simulate a traffic road network. The software is used for multimodal transportation planning with an integrated network model for private and public transport. The TP matrix is used as inputs to the PTV Visum simulation, and the outputs are the predicted traffic volume. The TP matrix is assigned according to the available traffic volumes. The input information from the PTV Visum [37] offers a guideline for the traffic flow completion model. Traffic information are collected each minute for five of the links. Figure 2 depicts a screenshot of the simulation showing connections 1–5 that are the highway links.



**Figure 2.** Case study traffic network with five highways links. The numbers 1 to 5 illustrate 5 highways links.



### 3. DBN for Time Series Forecasting

Machine learning modelling based on DBN has emerged as a technique to improve measurement data. DBNs are deep neural network models comprising of multiple layers of hidden nodes representing latent variables for detecting features extracted from the original multi-dimensional input data [38]. These models can be trained in a generative unsupervised manner where the model learns to probabilistically reconstruct the inputs from abstracted features extracted at each layer [38]. Following this learning step the DBN can be further trained as a discriminative supervised learning model to perform classification or time series prediction. There are three main reasons for using DBN as follows: They take numerous non-linear hidden layers, have the ability to be pre-trained in an unsupervised manner and allow the hidden state to be factored in an arbitrary way.

The traffic prediction algorithm is designed based on current and historical traffic flows data derived from a database of RSUs. We propose a strategy that predicts the activity time for every one of the 5 highways links over a brief time horizon in a transportation network which comprises of two stages: (1) traffic time information fulfilment and (2) Short-term traffic flow forecasting. In stage 1, trip distribution estimation is derived from the TP matrix to create traffic time information at each link based on demand and recorded information in the initial TP matrix. In stage 2, we utilize the traffic volume data at each link produced from stage 1 to anticipate traffic flow recursively by a network using two RBMs by adjusting in historical information to account for unpredictable changes. In this study, we designed a three-layer DBN constructed by using two stacked RBMs [25,26] to propose the traffic flow time series prediction model.

When high dimension data are input to the units of visible layer of an RBM, the units in the RBM's hidden layer detects the feature of data among different classes according to the connection weights. The connection of units of RBM is restricted to different layers, which means that no connections exists between the units of same layer, so the paired layers are termed as a restricted Boltzman machine. When the hidden layer of one RBM is used as a visible input layer for a second RBM, the second RBM's hidden layer determines "the feature of features" of the original input data. Therefore, the two stacked RBMs are able to determine a restricted set of features derived from the original higher dimensional input parameters.

In the initial step of the training process, the data units for each layer are set randomly to values of 0 or 1. For training the algorithm we randomly selected 30 training instances and evaluated the model on 30 test instances respectively. The weights  $w_{ij}$  among data units for each layer are set to values between 0 and 1. There is no connections between units of each layer of RBMs. The input units  $v_i$  of visible layer of RBMs are shown as  $x(t - \alpha)$ ,  $x(t - 2\alpha)$ ,  $\dots$ ,  $x(t - n\alpha)$  for input data  $x(t)$ ,  $t = 1, 2, 3, \dots, T$ . We calculate the expectation of data by  $p_{ij}^{data} = \langle x(t - i\alpha)h_j \rangle$ . Then, we calculate the expectation for reconstruction by  $p_{ij} = \langle v_i h_j \rangle$ , where  $h_j$  refers to the values of unites in the hidden layer of the RBM,  $\alpha$  is a positive integer and  $v_i$  is the binary state of input  $x(t)$ ,  $i = 1, 2, 3, \dots, n$  where  $n$  is defined as the dimension of the input data which shows the number of units on visible layer of RBMs. After this initial step, the weights  $w_{ij}$  should be updated by  $\Delta w_{ij} = \beta(p_{ij}^{data} - p_{ij})$ , where  $\beta$  is a rate of learning ( $0 < \beta < 1$ ). The hidden layer of first RBM then feeds in as the visible layer of second RBM. When the visible layer of each RBM receives the higher dimension information as inputs, the respective hidden layers classify the components of information among various classes using association weights. There is a limitation of association between units of each layer, so the matched layers are considered as RBM [27,39]. The hidden layer of the second RBM further evaluates the classified information to extract a reduced set of features. This initial training step ensures that RBM weights are approximated close to the ideal solution. The back-propagation (BP) algorithm [40] is then used for fine-tuning the weights of each RBM to get a refined prediction. Here the loss function used to evaluate the model's performance is based on the mean squared error (MSE) among  $x(t)$  and  $x(tn - \alpha)$ . The MSE is considered as a stopping criteria based on whether the MSE is small enough  $MSE < \varepsilon$ , where  $\varepsilon$  is a small and positive parameter. The stages of the model training process are shown in Figure 3.

In order to optimize DBN model further the number of units in the RBMs visible and hidden layers together with the learning rate are optimized by FFA according to the characteristic for neural network prediction models [41,42]. Figure 4 displays the diagram of the proposed DRBM-FFA modelling framework. Here the traffic detector data is utilized to describe present traffic patterns that is found in the historical database, whereby  $n$  best comparable patterns are chosen from historical data to derive the current traffic state and anticipate travel time using the proposed model.

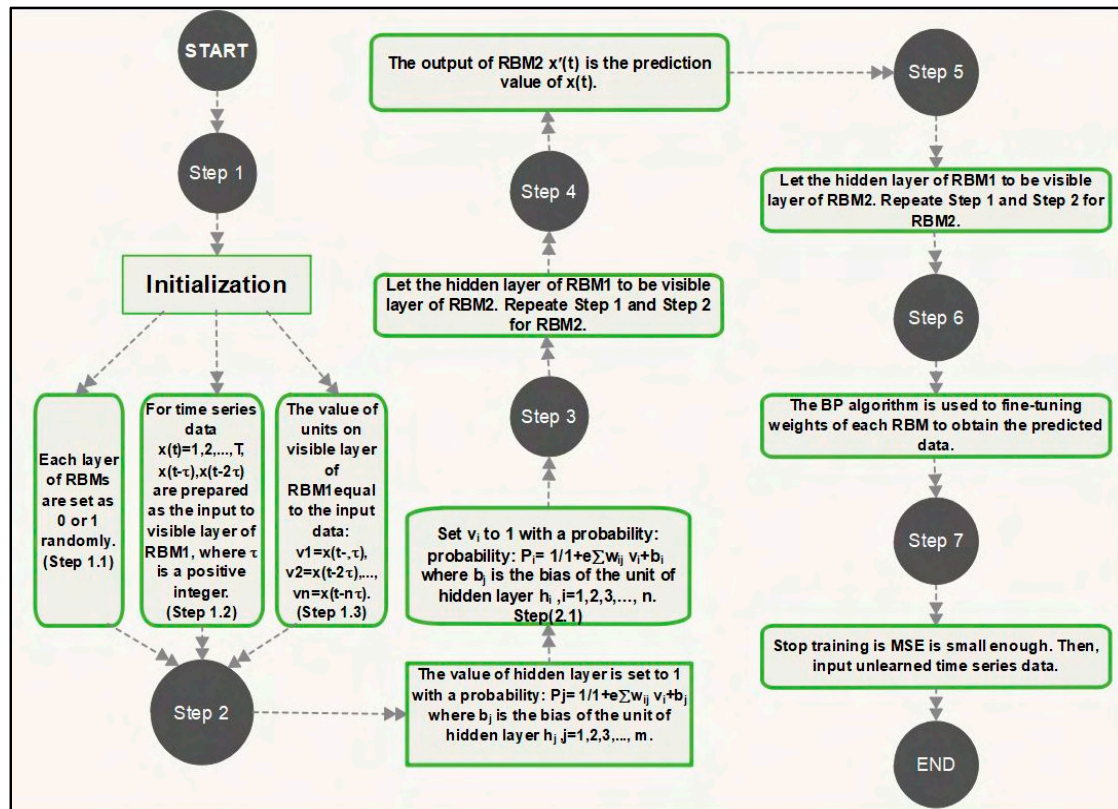


Figure 3. Steps of (DBN) with two (RBMs).

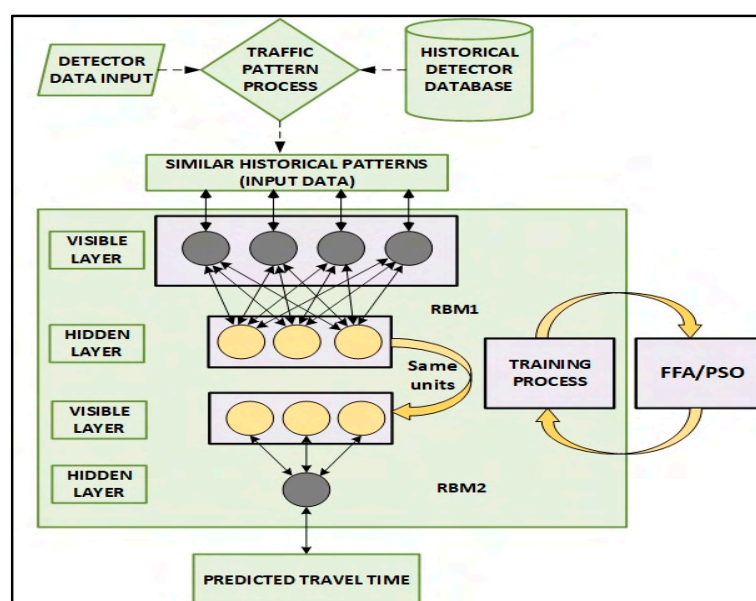


Figure 4. Proposed DRBM-FFA prediction model.

#### 4. Optimization of the DBN Prediction Model

Generally, the number of layers, number of units on input layer, hidden layer and learning rate should be optimized for designing an effective neural network model. Our model utilizes a three-layer DBN with the adoption of FFA [43] to find the learning rate of RBM, number of units on input layer and number of units on both hidden layers for RBM1 and RBM2 as shown in Figure 4.

##### *Optimization by Firefly Algorithm (FFA)*

This section clarifies the structure of the DBN parameter' optimization utilizing the FFA. The quantity of layers, the quantity of units within each layer, and the rate of learning of the DBN should be chosen to accurately model volume traffic flow prediction. The FFA is a nature-inspired optimization method based on the social mating behaviour of fireflies [43]. This algorithm belongs to the class of swarm intelligence techniques that is based on the bioluminescence flashing behaviour of fireflies, which acts as a signaling system to attract other fireflies and was developed by Yang [44]. In this algorithm each firefly can flash with some degree of brightness. This light can be attractive for other neighboring fireflies and their attraction is influenced by the distance between fireflies. Two fireflies which are close together will have higher attraction to each other. Each firefly symbolizes a point in a search space and the objective function is denoted by the attractiveness degree of each firefly. The fireflies should move towards their neighbours with the highest attraction. There are two essential parts to FFA: the difference of light intensity, and the definition of their engaging quality. It is considered that the attraction of each firefly is measured by its light intensity. The measurement is related to the encoded objective function. Note that the light intensity  $L(d)$  differs with the distance  $d$ , based on Equation (3):

$$L = L_0 e^{-\gamma d^2} \quad (3)$$

where intensity of light and value of the absorption coefficient are displayed by  $L_0$  and  $\gamma$  separately. As a firefly's attractive quality corresponds to the light intensity realized through neighboring fireflies. Then, we would be able to characterize the attraction  $\beta$  of a firefly as being:

$$\beta = \beta_0 e^{-\gamma d^2} \quad (4)$$

where  $\beta_0$  is the attraction at  $d = 0$ . The distance among any two fireflies  $i$  and  $j$  at  $X_i$  and  $X_j$  can be the Cartesian distance  $d_{ij} = \|X_i - X_j\|^2$  or the 2-norm.

If a firefly  $i$  is attracted to another brighter firefly  $j$  and firefly  $i$  is moving towards  $j$ , then the movement of a firefly  $i$  can be measured by following equation:

$$X_i = X_i + \beta_0 e^{-\gamma d^2} (X_j - X_i) + \alpha \varepsilon_i \quad (5)$$

where the second term will also contribute to the intensity of attraction and the third term comes from a Gaussian distribution which is presented by the vector of random variables  $\varepsilon_i$ .

We utilized the FFA to choose the best number of units per layer of RBM, assuming the prediction model is using two stacked RBMs as shown in Figure 4. The visible and hidden layers of the first RBM have  $n$  and  $m$  units respectively. The visible layer of the first RBM is similar to the visible layer of the second RBM. The hidden layer of the second RBM has 1 unit that is the output of the prediction model. Given the rate of learning of RBM is denoted  $\varepsilon$ , a firefly is intended to be represented as a vector  $X_i$  denoted as a three-dimension vector  $X(n; m; \varepsilon)$ , where  $n = 1, 2, \dots, n$ ; and  $m = 1, 2, \dots, m$ ; and  $\varepsilon \in (0, 1)$ . Using a population of fireflies with size  $P$ , the FFA algorithm can be used to improved prediction performance of DBN [13]. The FFA adopted to choose the ideal number of input and hidden units of DBN. The FFA is outlined as follows in Algorithm 1.



**Algorithm 1.** Firefly Algorithm

---

Objective function  $f(x)$ ,  $X_i = (x_1, x_1, \dots, x_1)^T$   
Decide the population size of fireflies  $P$  and set the iteration number of  $I$ .  
Initialize a population  $X_i (i = 1, 2, 3, \dots, n)$  of *fireflies*  
Outline  $\gamma$  as light absorption coefficient  
While ( $t < \text{Maximum Generation}$ )  
  For  $i = 1 : n$  all  $n$  *fireflies*  
  For  $j = 1 : i$  all  $n$  *firefiles*  
  Light intensity  $I_i$  at  $X_i$  is determined by  $f(X_i)$   
  Evaluate per firefly via the (MSE) among the predicted value  $\hat{x}(t)$  and original data  $x(t)$ .  
  If ( $I_j > I_i$ )  
  Move firefly  $i$  towards  $j$  in all  $d$  dimensions  
  End if  
  Attraction differs with distance  $r$  via  $\exp[-\gamma^r]$   
  Assess new solutions and update light intensity  
  End for  $j$   
  End for  $i$   
  Rank the fireflies and discover the current best  
  Find the best firefly with best attraction from its history  
End *while*

---

**5. Forecasting Results**

The time series data was used in short term prediction simulation to evaluate the performance of the proposed DRBM-FFA approach. To objectively benchmark the performance accuracy of the proposed approach it was compared with two other well know predictive modelling approaches namely a conventional Multi-Layer Perceptron (MLP) neural network and a linear ARIMA model [45]. To also have a fair evaluation of the hybrid FFA optimization strategy used, hybrid variants of MLP optimized using FFA (MLP-FFA) and ARIMA optimized using particle swarm optimization (ARIMA-PSO) were also compared with the proposed DRBM-FFA method. Figure 5 shows the designing an optimized predictor of DBN via the best firefly.

Table 2 lists the parameters and values for each of the algorithms evaluated in our prediction experiments.

**Table 2.** The used parameters in the prediction experiments.

| Description                  | Model Elements/Parameters | Quantity         |
|------------------------------|---------------------------|------------------|
| Population of PSO            | P                         | 10               |
| The number of RBM            | RBM1,RBM2                 | 2                |
| The number of input layer    | $N (1 \leq N \leq 20)$    | Given by FFA     |
| Absorption coefficient       | $\gamma$                  | 0.1              |
| Velocity coefficient         | $c_1, c_2$                | 1.0              |
| The number of hidden layer   | $M (1 \leq M \leq 20)$    | Given by FFA     |
| The number of output         | -                         | 1                |
| Interval of input data       | $\tau$                    | 1                |
| Learning rate of RBM         | $\epsilon$ (Step 2)       | Given by FFA     |
| Learning rate of BP          | -                         | Given by FFA     |
| Population of FFA            | P                         | 10               |
| Iteration times of BP        | L                         | $100 < L < 5000$ |
| Biases of units              | $b_i, b_j$                | 0.0              |
| Convergence parameter of BP  | $\alpha$                  | 0.05             |
| Convergence parameter of RBM | $\beta$                   | 0.0005           |
| Convergence period of RBM    | k                         | 50               |

Short-term prediction accuracy of the DRBM-FFA model compared against the ARIMA, MLP-FFA and ARIMA-PSO are shown in Figure 6. Each algorithm is used to predict traffic flows in all five links in the traffic network where traffic data is utilized to predict traffic flow for the whole transportation network. The short-term prediction precision of the DRBM-FFA is compared against each of the other models and the results are shown in Figures 7–10.

The flowchart of the processing steps is shown in Figure 5.

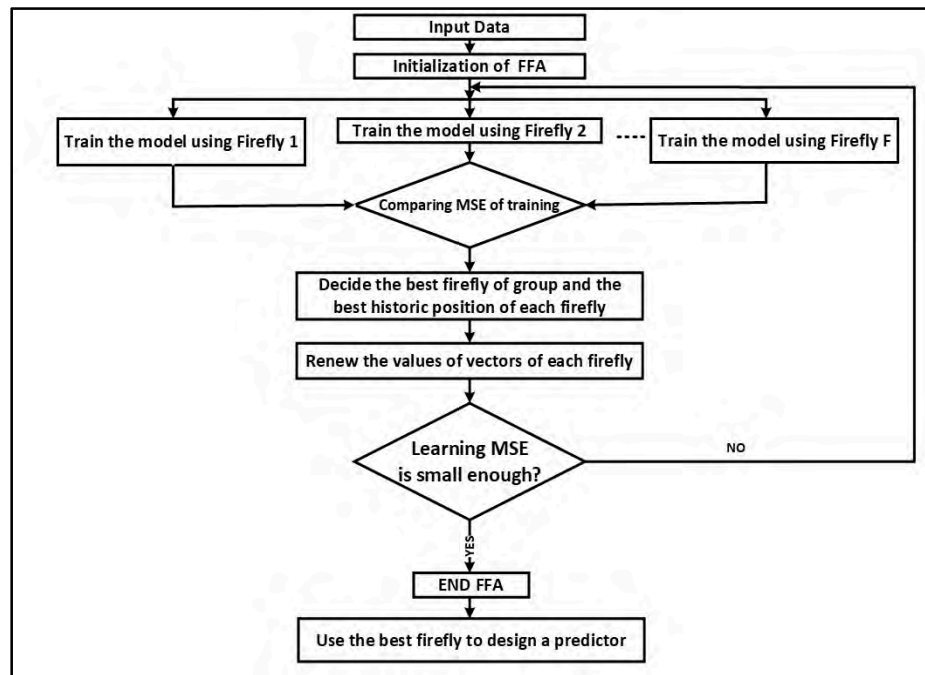


Figure 5. Designing an optimized predictor of DBN via the best firefly.

Different statistical estimators are applied to assess the performance of the proposed DRBM-FFA method. These estimators are as follows: the (MSE) shown in Equation (6), the coefficient of determination ( $R^2$ ) presented in Equation (7), the root mean square (RMSE) presented in Equation (8), correlation coefficient ( $r$ ) presented in Equation (9), mean absolute percentage error (MAPE) shown in Equation (10), root-mean square percentage error (RMSPE) displayed in Equation (11).

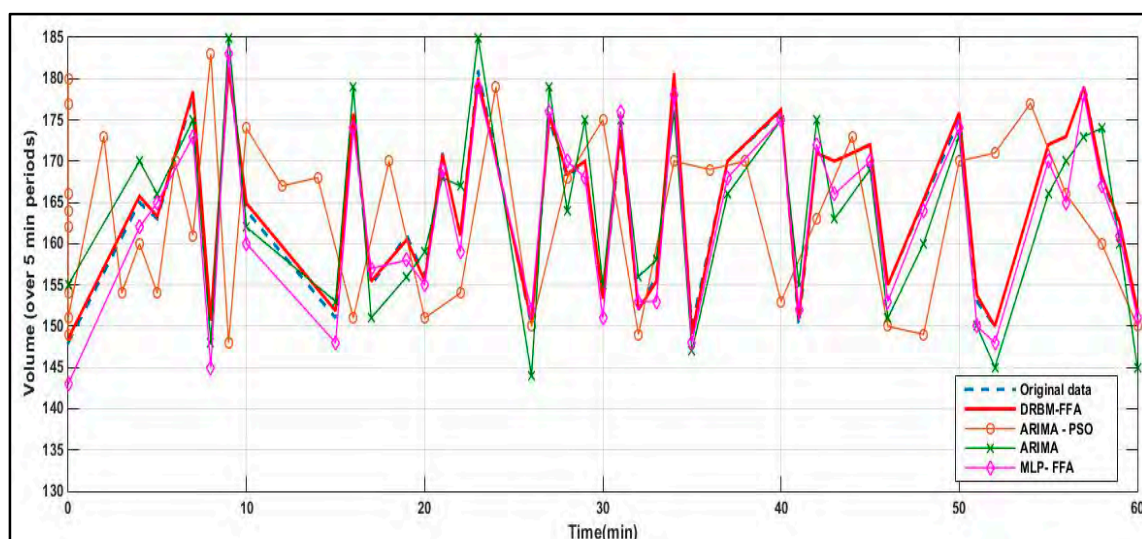


Figure 6. Prediction results by DRBM-FFA, ARIMA-PSO, ARIMA and MLP-FFA methods.

$$\text{MSE} = \frac{1}{r} \sum_{i=1}^r (D_{pi} - D_{ai})^2 \quad (6)$$

$$R^2 = 1 - \frac{\sum_{i=1}^r (D_{pi} - D_{ai})^2}{\sum_{i=1}^r (D_{pi} - D_{av})^2} \quad (7)$$

$$\text{RMSE} = \sqrt{\frac{1}{r} \sum_{i=1}^r (D_{pi} - D_{ai})^2} \quad (8)$$

$$r = \frac{\sum_{i=1}^n (D_{pi} - \overline{D_{pi}}) \cdot (D_{ai} - \overline{D_{ai}})}{\sqrt{\sum_{i=1}^n (D_{pi} - \overline{D_{pi}})^2 \cdot \sum_{i=1}^n (D_{ai} - \overline{D_{ai}})^2}} \quad (9)$$

$$\text{MAPE} = \frac{1}{r} \sum_{i=1}^r \left| \frac{D_{pi} - D_{ai}}{D_{ai}} \right| \times 100 \quad (10)$$

$$\text{RMSPE} = \sqrt{\frac{1}{n} \sum_{l=1}^n \left[ \frac{\overline{D_{pl}} - D_l}{D_l} \right]^2} \quad (11)$$

where  $n$  is the quantity of data,  $D_{pi}$  is the predicted value;  $D_{av}$  is the average of the actual values;  $D_{ai}$  is the actual value;  $\overline{D_{pl}}$  is the predicted traffic flow;  $D_l$  shows the measured traffic flow for link  $l$ ;  $\overline{D_{pi}}$  and  $\overline{D_{ai}}$  are the mean value of  $D_{pi}$  and  $D_{ai}$ , respectively. The coefficient of determination,  $R^2$  represents the linear regression line among the predicted values of the neural network model. The essential output, is applied as a measure of performance. Expressed differently,  $R^2$  is the square of the correlation between the response values and the predicted response values. The closer  $R^2$  is to 1, the better the model can fit the actual data [46]. This measurement controls the degree of success the fit has in stating the change of the data. It can be indicated as the square of the multiple correlation coefficients, and the coefficient of multiple determinations. The smaller amount of MAPE has a superior performance model, and conversely, in the case of  $r$ . The detail prediction errors (MSEs) for the original data are shown in Table 3.

**Table 3.** The detail prediction errors (MSEs).

| Structure and Evaluation  | MLP-FFA | ARIMA  | ARIMA-PSO | DRBM-FFA |
|---------------------------|---------|--------|-----------|----------|
| Learning rates            | 0.85    | 0.64   | 0.73      | 0.98     |
| Iterations                | 336     | 350    | 298       | 200      |
| Learning MSE              | 109.21  | 122.4  | 108.9     | 98.70    |
| Short-term prediction MSE | 234.38  | 280.50 | 126.11    | 109.38   |

Table 3 shows that the DRBM was able to outperform in comparison to the other approaches based on obtaining the lowest learning MSE and short-term prediction MSE based on the time series results shown in Figures 7–10. The MLP with FFA obtained the next lowest learning MSE and short-term prediction MSE followed by the ARIMA. The Monte Carlo method was used to acquire a more objective evaluation of the performance of each approach that is based on sampling testing data based on sub-blocks to evaluate the forecasting efficiency of the algorithm.

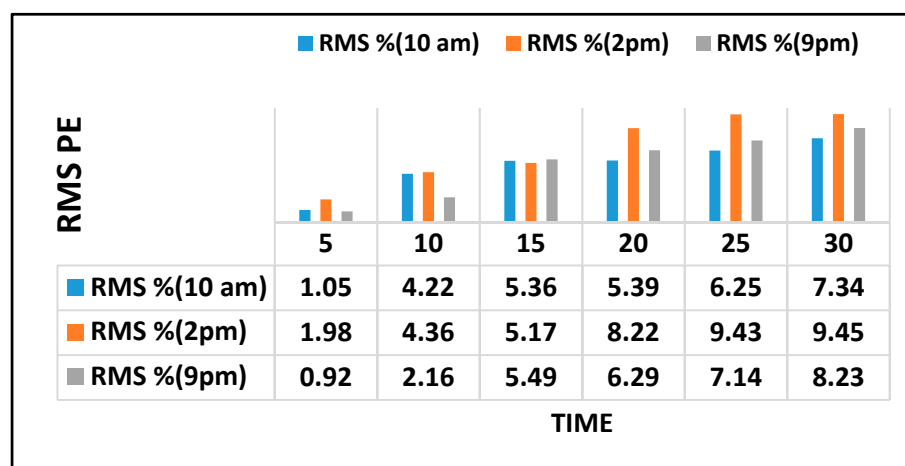
Experiments to determine traffic flow prediction performance over five time horizons were carried out to evaluate the performance of the MLP-FFA, ARIMA, ARIMA-PSO and DRBM-FFA methods. Let  $P(t+1)$  represent the estimated flow for the  $(i+1)$ th time interval in the future. For the first forecasting interval  $(i+0)$ , the flow is represented by  $P(t)$ . Table 4 shows the forecasting outcomes from the test data for 5 links. In Table 4, each “ $t$ ”, “ $t+1$ ”, “ $t+2$ ” and “ $t+3$ ” is a 15-min interval into the future. The results show that the performance of all four prediction models improves when forecasting further into the future. Values in bold style show the minimum quantities for RMSE,  $r$  and MAPE.

Table 4 shows that all error measurement for DRBM-FFA are less than those for the other algorithms for all 15-min prediction intervals. As shown in Table 4, DRBM-FFA outperformed MLP-FFA, ARIMA, and ARIMA-PSO forecasters for all three time intervals. As anticipated, the PSO improved prediction accuracy of the ARIMA model.

Figure 7 further illustrates the prediction results of selected links every 5 min for DRBM-FFA for the next 30 min which was determined using root-mean square percentage Error (RMSPE). Figure 7 demonstrates the RMSPE for the selected links.

**Table 4.** Traffic flow prediction results.

| Predictor | Time Interval | r   | RMSE | MAPE   |
|-----------|---------------|-----|------|--------|
| MLP-FFA   | t             | 3.2 | 6.8  | 12.07% |
|           | t + 1         | 3.5 | 7.2  | 13.95% |
|           | t + 2         | 3.6 | 7.8  | 14.89% |
|           | t + 3         | 3.9 | 7.9  | 15.32% |
| ARIMA     | t             | 4.4 | 9.1  | 13.56% |
|           | t + 1         | 4.6 | 9.7  | 15.37% |
|           | t + 2         | 6.8 | 14.2 | 18.93% |
|           | t + 3         | 8.5 | 15.7 | 23.24% |
| ARIMA-PSO | t             | 3.3 | 6.8  | 9.39%  |
|           | t + 1         | 3.4 | 6.9  | 9.89%  |
|           | t + 2         | 3.7 | 7.2  | 10.48% |
|           | t + 3         | 3.9 | 7.8  | 11.57% |
| DRBM-FFA  | t             | 2.9 | 6.1  | 8.75%  |
|           | t + 1         | 3.1 | 6.4  | 9.63%  |
|           | t + 2         | 3.4 | 6.9  | 10.31% |
|           | t + 3         | 3.5 | 7.1  | 11.12% |



**Figure 7.** RMSPE for the targeted links.

In addition to the given experiments, Monte Carlo [47] method is applied to assess the sensitivity and accuracy of each predictive algorithm due to the stochastic variation of traffic data. Firstly, in each experiment, the ratio of traffic flow for links is calculated. Secondly, 50% of the ratio of traffic flow for links are designated randomly. Thirdly, selected data is increased by a Gaussian random variable  $r$ . Fourthly, the new ratio of traffic flow for each link are served to the predictive method, and the results are recorded. The final stage is where, the four previous stages should be repeated 1000 times per data sample. Hence, the standard deviation of the Monte Carlo results are calculated and the coefficient  $r$  is supposed to be a Gaussian random variable  $r \sim N(1, 0.1)$ . Figure 8 shows the error for a particular data sample for our prediction model.



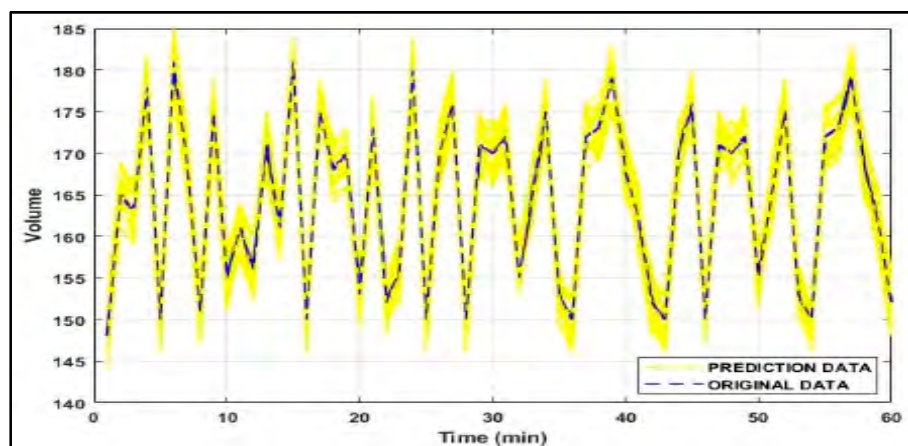


Figure 8. Monte Carlo experiment results over a 60-min period.

Calculating the computation time (CT) taken for algorithm completion is especially necessary in the real-time IoV applications when OBU computer systems should quickly respond to any external occurrences. In this work, to obtain a fair comparison, the same computer is used for measuring the computation time (as measured by the “tic–toc” MATLAB function) [48,49]. The MATLAB R2017b on an Intel Core i5 laptop with Windows 10 system is used to carry out this measurement. In this study, the stopwatch timer functions, tic and toc, are used to calculate the computation time. Invoking tic starts the timer, and the next toc reads the elapsed time in MATLAB. The CPU time returns the total CPU time (in seconds). The line graph compares the amount of computation time on the DRBM-FFA, MLP-FFA, ARIMA, and ARIMA-PSO in 30 runs. When comparing the data resulting from the plot, the average time needed for ARIMA-PSO and MLP-FFA calculation is approximately 0.6 (s). ARIMA has a high computation time of about 0.7 (s). In contrast, DRBM-FFA has the lowest computation time of 0.5 (s). Figure 9 shows changes in the computation time between the DRBM-FFA, MLP-FFA, ARIMA, and ARIMA-PSO methods.

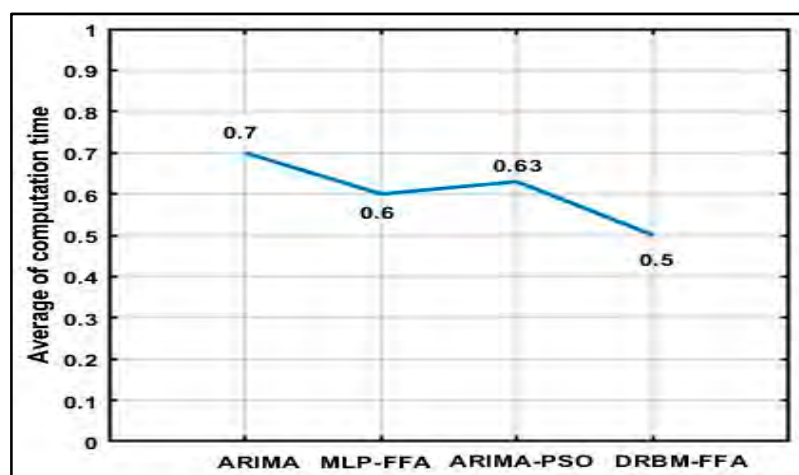


Figure 9. Average of computation time.

The standard deviation (SD) of the function values achieved in the experimental trials were also evaluated to determine prediction stability. Here the same initial conditions are used for all algorithms. The results in Figure 10 show that the standard deviation of the function values of all the other algorithms have larger values in comparison to the proposed DRBM-FFA approach. ARIMA-PSO is the next most robust approach, followed by MLP-FFA and ARIMA. The average of standard deviation for ARIMA is clearly the highest with values at 0.050, and hence their solutions qualities can be deduced to be less stable.

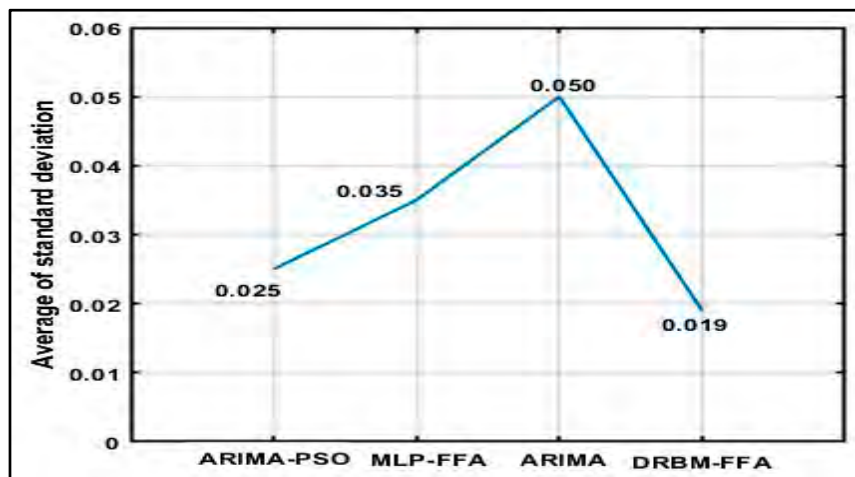


Figure 10. Average of standard deviation.

In sum, the average of standard deviation obtained by DRBM-FFA is lower than those obtained by MLP-FFA, ARIMA, and ARIMA-PSO, so the proposed hybrid is very robust. Also, the swarm method of PSO achieves better result in standard deviation by ARIMA-PSO than MLP-FFA, ARIMA. The less the operating cost can be achieved with smaller standard deviations.

To further test the effectiveness of the proposed scheme, the proposed DRBM-FFA algorithm is compared in terms of complexity with the MLP-FFA, ARIMA, ARIMA-PSO algorithms. The computational complexity of the proposed model depends on the number of training samples in datasets, the structure of the RBM, the time complexity for the initialization of the fireflies, calculating the fitness values, attraction mechanism and updating the light intensity for selecting the best firefly. So, the overall computational complexity is determined as follows:

$$\begin{aligned}
 O(\text{RBM}, \text{FFA}) &= O(\text{RBM}) + O(\text{FFA}) \\
 &= O(\text{RBM}) + O(\text{initialization}) + O(\text{cost function calculation}) \\
 &\quad + O(\text{attraction}) + O(\text{light intensity}) + O(\text{best firefly selection})
 \end{aligned} \quad (12)$$

The time complexity for the initialization of the fireflies is  $O(SN)$ ; where the  $SN$  is the maximum size of the population in FFA. The time complexity for calculating the fitness values is  $O(SN)$ ; the time complexity for updating the light intensity extreme value is  $O(SN)$ ; the time complexity for selecting the best firefly extreme value is  $O(SN)$ . Therefore, the worst time complexity of FFA for one iteration is:  $O(SN) + O(SN) + O(SN) + O(SN) + O(SN)$ , which can be simplified as  $O(\text{FFA}) = O(SN)$ . The fixed population size ( $SN$ ) is only considered to have an influence on the time complexity for DRBM-FFA, MLP-FFA and ARIMA-PSO models.

The overall computational complexity of an RBM depends on number of hidden units, number of the data points and number of output nodes. Additionally the computational complexity of RBM for the sequential searching takes  $f(T)$  times to get the subset of failed patterns in the best case. Therefore, the final computational complexity of the proposed method is as follows:

$$O(\text{RBM}, \text{FFA}) = O(SN + (f(T) + f(T) \times n_h + o)) \quad (13)$$

where,  $T$  is the number of data points,  $n_h$  is number of hidden nodes,  $o$  is the number of output nodes.

For the MLP-FFA model, the computational complexity of an MLP depends on hidden nodes, number of outputs, and the number of training samples  $t$ . Given the computational complexity of FFA is  $O(SN)$ , the final computational complexity of the (MLP-FFA) is as follows:

$$O(\text{MLP}, \text{FFA}) = O(SN + (t(h + o))) \quad (14)$$

where  $t$  is the number of training samples,  $h$  is number of hidden nodes,  $o$  is the number of output nodes.

For the ARIMA-PSO model, the measure of complexity depends on the order values for  $p$  and  $q$  in ARIMA, and the time complexity of PSO algorithm. For the PSO algorithm, the time complexity can be obtained as follows: The time complexity for the initialization of the particle swarm is  $O(SN)$ ; the time complexity for calculating the fitness values is  $O(SN)$ ; the time complexity for updating the individual extreme value is  $O(SN)$ ; the time complexity for selecting the best individual extreme value is  $O(SN)$ ; the time complexity for updating the velocities and positions is  $O(SN)$ . Therefore, the worst time complexity of PSO algorithm for one iteration is:  $O(SN) + O(SN) + O(SN) + O(SN) + O(SN)$ , which can be simplified as  $O(PSO) = O(SN)$ . The time complexity for ARIMA ( $p, d, q$ ) depends on both  $p$  and  $q$  values [50]. Computation of AR and MA coefficients thus takes  $O((N - p) p^2)$  and  $O((N - q) q^2)$  time, respectively, where  $N$  is the length of historical values. So, the total time of the ARIMA model is as follows:

$$O(\text{ARIMA}) = O((N - p) \times p^2 + (N - q) \times q^2) \quad (15)$$

We can easily see that the complexity of ARIMA grows significantly as we consider higher order values for  $p$  and  $q$ . Therefore, the final computational complexity of the (ARIMA-PSO) is as follows:

$$O(\text{ARIMA}, \text{PSO}) = O(SN + (N - p) \times p^2 + (N - q) \times q^2) \quad (16)$$

It can be concluded that  $O(\text{ARIMA}) > O(\text{ARIMA}, \text{PSO}) > O(\text{MLP}, \text{FFA}) \simeq (\text{DRBM}, \text{FFA})$  which suggests that the hybrid approaches give an improvement in both the training and prediction process.

The effect of scalability on the computational complexity of the DRBM-FFA algorithm was also analyzed. For this purpose, time complexity of the DRBM-FFA algorithm for solving the Rosenbrock function with different dimensions was calculated as described in [51]. The Rosenbrock function was selected because it interacts between the candidate algorithms parameters. To establish the time complexity, an after-code execution time ( $T_0$ ) and the execution time of Rosenbrock function for 200,000 evaluations ( $T_1$ ) were measured and the mean of five execution times for DRBM-FFA on the Rosenbrock function using 200,000 evaluations ( $\hat{T}_2$ ) was measured. A new population is formed by the fireflies, in search of neighborhoods of the chosen solutions based on their quality. As the number of fireflies is same as  $SN$ , in each cycle, the fireflies carry out  $SN$  searches. Thus, when the maximum cycle number (MCN) is achieved, a total of  $SN \times \text{MCN}$  searches are performed. After that, the complexity of the algorithm is established using  $\frac{(\hat{T}_2 - T_1)}{T_0}$  as shown in Table 5 which displays the increase of  $\hat{T}_2$  by less than a factor of increment dimension. As a result, it is noted that the FFA algorithm time complexity is not excessively dependent on the problem dimension and scales at  $O(n)$ . Specifically, the DRBM-FFA totally outperforms the other methods based on aspects of solution quality and running time.

**Table 5.** Time Complexity Comparison.

| Model     | D   | $T_0$ | $T_1$ | $\hat{T}_2$ | $\frac{(\hat{T}_2 - T_1)}{T_0}$ |
|-----------|-----|-------|-------|-------------|---------------------------------|
| MLP-FFA   | 10  | 0.490 | 0.465 | 2.809       | 4.780                           |
|           | 50  | 0.491 | 0.643 | 2.911       | 4.610                           |
|           | 100 | 0.493 | 0.720 | 3.108       | 4.800                           |
| ARIMA     | 10  | 0.389 | 0.509 | 3.142       | 6.722                           |
|           | 50  | 0.378 | 0.734 | 3.708       | 7.855                           |
|           | 100 | 0.398 | 0.821 | 3.698       | 7.212                           |
| ARIMA-PSO | 10  | 0.470 | 0.489 | 2.809       | 4.852                           |
|           | 50  | 0.489 | 0.631 | 2.902       | 4.637                           |
|           | 100 | 0.489 | 0.715 | 3.212       | 5.090                           |
| DRBM-FFA  | 10  | 0.411 | 0.233 | 0.703       | 1.145                           |
|           | 50  | 0.412 | 0.474 | 1.336       | 2.046                           |
|           | 100 | 0.412 | 0.732 | 1.994       | 2.857                           |

As a result, it is noted that the DRBM-FFA time complexity is not excessively dependent on the problem dimension. Specifically, the DRBM-FFA totally outperforms the others based on aspects of solution quality and the running time.

## 6. Conclusions

The precise prediction of traffic state can help to reduce traffic congestion in large metropolitan areas. In this study, we predicted short term traffic flows using links in a traffic network and traffic data. The aim of this study was to design an accurate method for traffic flow prediction over various time periods which takes into account the spatial features of a road network to determine not only the distance but also the average speed of traffic on the links. Therefore, we proposed a DRBM-FFA prediction model based on a DBN comprising two stacked RBMs using a pre-training and fine-tuning learning algorithm. This was combined with FFA for optimizing the model's parameters against its performance on training and testing data. Historical data on road traffic flow was used to construct and evaluate the prediction model. Results showed that for all the three evaluation measures ( $r$ , RMSE and MAPE), the proposed model has better performance in comparison with other traffic volume prediction methods and is able to accurately predict the traffic flow.

**Author Contributions:** Conceptualization, S.G., S.A.S., M.H.A., F.D.; Methodology, S.G.; Software, S.G. and S.A.S.; Validation, S.G., M.H.A. and F.D.; Formal Analysis, S.G. and S.A.S.; Investigation, S.G., S.A.S. and M.H.A.; Resources, S.G.; Data Curation, S.G. and S.A.S.; Writing-Original Draft Preparation, S.G.; Writing-Review & Editing, M.H.A., F.D. and M.N.K.; Visualization, S.G.; Supervision, M.N.K. and M.H.A.; Project Administration, S.G. and M.N.K. and M.H.A.; Funding Acquisition, M.N.K.

**Funding:** This research was funded by [University Teknologi Malaysia] grant number [01M16].

**Conflicts of Interest:** The authors declare no conflict of interest.

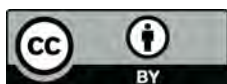
## References

1. Soleymani, S.A.; Abdullah, A.H.; Zareei, M.; Anisi, M.H.; Vargas-Rosales, C.; Khan, M.K.; Goudarzi, S. A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing. *IEEE Access* **2017**, *5*, 15619–15629. [[CrossRef](#)]
2. Soleymani, S.A.; Abdullah, A.H.; Anisi, M.H.; Altameem, A.; Hasan, W.H.; Goudarzi, S.; Mandala, S.; Razak, Z.B.; Noor, N.M. BRAIN-F: Beacon rate adaption based on fuzzy logic in vehicular ad hoc network. *Int. J. Fuzzy Syst.* **2017**, *19*, 301–315. [[CrossRef](#)]
3. Ghafoor, K.Z.; Bakar, K.A.; Lloret, J.; Khokhar, R.H.; Lee, K.C. Intelligent beaconless geographical forwarding for urban vehicular environments. *Wirel. Netw.* **2013**, *19*, 345–362. [[CrossRef](#)]
4. Ghafoor, K.Z.; Mohammed, M.A.; Lloret, J.; Bakar, K.A.; Zainuddin, Z.M. Routing protocols in vehicular ad hoc networks: Survey and research challenges. *Netw. Protocols Algorithms* **2013**, *5*, 39–83. [[CrossRef](#)]
5. Abdulhai, B.; Porwal, H.; Recker, W. Short-term traffic flow prediction using neuro-genetic algorithms. *J. Intell. Transp. Syst.* **2002**, *7*, 3–41. [[CrossRef](#)]
6. Vlahogianni, E.I.; Karlaftis, M.G.; Golias, J.C. Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. *Transp. Res. Part C Emerg. Technol.* **2005**, *13*, 211–234. [[CrossRef](#)]
7. Williams, B.M. Multivariate vehicular traffic flow prediction: Evaluation of ARIMAX modeling. *Transp. Res. Rec.* **2001**, *1776*, 194–200. [[CrossRef](#)]
8. Williams, B.M.; Hoel, L.A. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *J. Transp. Eng.* **2003**, *129*, 664–672. [[CrossRef](#)]
9. Sun, S.; Zhang, C.; Yu, G. A Bayesian network approach to traffic flow forecasting. *IEEE Trans. Intell. Transp. Syst.* **2006**, *7*, 124–131. [[CrossRef](#)]
10. Dougherty, M.S.; Cobbett, M.R. Short-term inter-urban traffic forecasts using neural networks. *Int. J. Forecast.* **1997**, *13*, 21–31. [[CrossRef](#)]
11. Ledoux, C. An urban traffic flow model integrating neural networks. *Transp. Res. Part C Emerg. Technol.* **1997**, *5*, 287–300. [[CrossRef](#)]
12. Shmueli, D. Applications of neural networks in transportation planning. *Prog. Plan.* **1998**, *50*, 141–204. [[CrossRef](#)]



13. Dia, H. An object-oriented neural network approach to short-term traffic forecasting. *Eur. J. Oper. Res.* **2001**, *131*, 253–261. [[CrossRef](#)]
14. Yin, H.B.; Wong, S.C.; Xu, J.M.; Wong, C.K. Urban traffic flow prediction using a fuzzy-neural approach. *Transp. Res. Part C Emerg. Technol.* **2002**, *10*, 85–98. [[CrossRef](#)]
15. Lan, L.W.; Huang, Y.C. A rolling-trained fuzzy neural network approach for freeway incident detection. *Transportmetrica* **2006**, *2*, 11–29. [[CrossRef](#)]
16. El Faouzi, N.-E. Nonparametric traffic flow prediction using kernel estimator. In Proceedings of the 13th International Symposium on Transportation and Traffic Theory, Lyon, France, 24–26 July 1996; pp. 41–54.
17. Smith, B.L.; Williams, B.M.; Oswald, R.K. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transp. Res. Part C Emerg. Technol.* **2002**, *10*, 303–321. [[CrossRef](#)]
18. Huisken, G. Soft-computing techniques applied to short-term traffic flow forecasting. *Syst. Anal. Model. Simul.* **2003**, *43*, 165–173. [[CrossRef](#)]
19. Qu, L.; Li, L.; Zhang, Y.; Hu, J. PPCA based missing data imputation for traffic flow volume: A systematical approach. *IEEE Trans. Intell. Transp. Syst.* **2009**, *10*, 512–522.
20. Chen, C.; Wang, Y.; Li, L.; Hu, J.; Zhang, Z. The retrieval of intra-day trend and its influence on traffic prediction. *Transp. Res. Part C Emerg. Technol.* **2012**, *22*, 103–118. [[CrossRef](#)]
21. Smith, B.; Scherer, W.; Conklin, J. Exploring imputation techniques for missing data in transportation management systems. *Transp. Res. Rec.* **2003**, *1836*, 132–142. [[CrossRef](#)]
22. Kalhor, S.; Anisi, M.; Haghighat, A.T. A new position-based routing protocol for reducing the number of exchanged route request messages in Mobile Ad-hoc Networks. In Proceedings of the IEEE Second International Conference on Systems and Networks Communications (ICSNC 2007), Cap Esterel, France, 25–31 August 2007; p. 13.
23. Sun, S.; Huang, R.; Gao, Y. Network-scale traffic modeling and forecasting with graphical lasso and neural networks. *J. Transp. Eng.* **2012**, *138*, 1358–1367. [[CrossRef](#)]
24. RAND. *Moving Los Angeles: Short-Term Transportation Policy Options for Improving Transportation*; RAND Corporation: Santa Monica, CA, USA, 2008.
25. *Traffic Choices Study: Summary Report*; Federal Highway Administration: Seattle, WA, USA, 2008.
26. Mamuneas, T.P.; Nadri, M.I. *Contribution of Highway Capital to Industry and National Productivity Growth*; Transportation Research Board: Washington, DC, USA, 1996.
27. Ackley, D.H.; Hinton, G.E.; Sejnowski, T.J. A learning algorithm for Boltzmann machines. *Cogn. Sci.* **1985**, *9*, 147–169. [[CrossRef](#)]
28. Hinton, G.E.; Sejnowski, T.J. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*; Rumelhart, D.E., McClelland, J.L., Eds.; MIT Press: Cambridge, MA, USA, 1986; Volume 1.
29. Hinton, G.E.; Osindero, S.; The, Y.W. A faster learning algorithm for deep belief nets. *Neural Comput.* **2006**, *1*, 1527–1544. [[CrossRef](#)] [[PubMed](#)]
30. van Lint, J.W.C.; Hoogendoorn, S.P.; van Zuylen, H.J. Accurate freeway travel time prediction with state-space neural networks under missing data. *Transp. Res. Part C Emerg. Technol.* **2005**, *13*, 347–369. [[CrossRef](#)]
31. Sun, S.; Zhang, C.; Yu, G.; Lu, N.; Xiao, F. Bayesian network methods for traffic flow forecasting with incomplete data. In *European Conference on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 419–428.
32. Lawrence, M.J.; Edmundson, R.H.; O'Connor, M.J. The accuracy of combining judgmental and statistical forecasts. *Manag. Sci.* **1986**, *32*, 1521–1532. [[CrossRef](#)]
33. Gao, Y.; Er, M.J. NARMAX time series model prediction: Feedforward and recurrent fuzzy neural network approaches. *Fuzzy Sets Syst.* **2005**, *150*, 331–350. [[CrossRef](#)]
34. Zheng, W.; Lee, D.; Shi, Q. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *J. Transp. Eng.* **2006**, *132*, 114–121. [[CrossRef](#)]
35. Sun, L.; Wu, Y.; Xu, J.; Xu, Y. An RSU-assisted localization method in non-GPS highway traffic with dead reckoning and V2R communications. In Proceedings of the IEEE 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), Yichang, China, 21–23 April 2012; pp. 149–152.
36. Box, G.E.; Jenkins, G.M.; Reinsel, G.C.; Ljung, G.M. *Time Series Analysis: Forecasting and Control*; John Wiley & Sons: New York, NY, USA, 2015.

37. PTV Group. Available online: <http://vision-traffic.ptvgroup.com/en-us/products/ptv-visum/> (accessed on 17 February 2018).
38. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [[CrossRef](#)] [[PubMed](#)]
39. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [[CrossRef](#)] [[PubMed](#)]
40. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representation by back-propagating errors. *Nature* **1986**, *323*, 533–536. [[CrossRef](#)]
41. Zhang, G.P. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* **2003**, *50*, 159–175. [[CrossRef](#)]
42. Gardner, E.; McKenzie, E. Seasonal exponential smoothing with damped trends. *Manag. Sci.* **1989**, *35*, 372–376. [[CrossRef](#)]
43. Łukasik, S.; Żak, S. Firefly algorithm for continuous constrained optimization tasks. In *Computational Collective Intelligence Semantic Web, Social Networks and Multiagent Systems*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 97–106.
44. Yang, X.S. *Nature-Inspired Metaheuristic Algorithms*; Luniver Press: Beckington, UK, 2010.
45. Box, G.E.P.; Jenkins, G. *Time Series Analysis, Forecasting and Control*; Cambridge University Press: Cambridge, UK, 1976.
46. Goudarzi, S.; Hassan, W.H.; Anisi, M.H.; Soleymani, S.A.; Shabanzadeh, P. A Novel Model on Curve Fitting and Particle Swarm Optimization for Vertical Handover in Heterogeneous Wireless Networks. *Math. Probl. Eng.* **2015**. [[CrossRef](#)]
47. Reuven, Y.R. *Simulation and the Monte Carlo Method*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1981; 304p.
48. Tok, D.S.; Shi, Y.; Tian, Y.; Yu, D.L. Factorized f-step radial basis function model for model predictive control. *Neurocomputing* **2017**, *239*, 102–112.
49. Lucon, P.A.; Donovan, R.P. An artificial neural network approach to multiphase continua constitutive modeling. *Compos. Part B Eng.* **2007**, *38*, 817–823. [[CrossRef](#)]
50. Gavirangaswamy, V.B.; Gupta, G.; Gupta, A.; Agrawal, R. Assessment of ARIMA-based prediction techniques for road-traffic volume. In Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems, Luxembourg, 28–31 October 2013; pp. 246–251.
51. Chai-Ead, N.; Aungkulanon, P.; Luangpaiboon, P. Bees and firefly algorithms for noisy non-linear optimisation problems. In Proceedings of the International Multi Conference of Engineering and Computer Scientists, Hong Kong, China, 16–18 March 2011; Volume 2.



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).